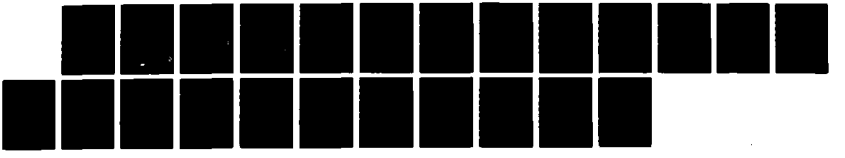
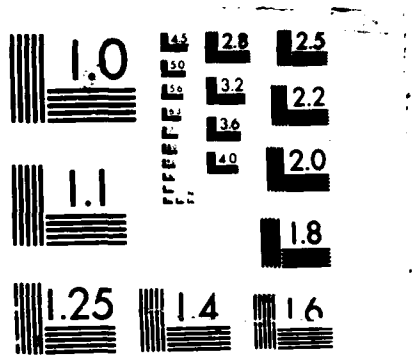
AD-A180 342 A MATROID ALGORITHM AND ITS APPLICATION TO THE EFFICIENT SOLUTION OF TWO.. (U) CARNEGIE-MELLON UNIV PITTSBURGH PA MANAGEMENT SCIENCES RESEAR.. 1/1

UNCLASSIFIED C BREZOVEC ET AL. FEB 87 N00014-85-K-0198 F/G 12/2 NL





(12)

A MATROID ALGORITHM AND ITS APPLICATION TO THE EFFICIENT SOLUTION OF TWO OPTIMIZATION PROBLEMS ON GRAPHS

by

Carl Brezovec*
Gerard Cornuejols*
and
Fred Glover**

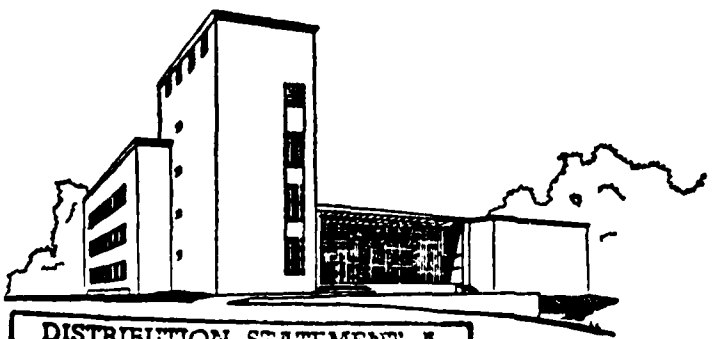
February, 1987

Carnegie-Mellon University

PITTSBURGH, PENNSYLVANIA 15213

GRADUATE SCHOOL OF INDUSTRIAL ADMINISTRATION

WILLIAM LARIMER MELLON, FOUNDER



(12)

W.P. No. 29-86-87

Management Science Research Report MSRR 534

A MATROID ALGORITHM AND ITS APPLICATION TO THE EFFICIENT SOLUTION OF TWO OPTIMIZATION PROBLEMS ON GRAPHS

by

Carl Brezovec*
Gerard Cornuejols*
and
Fred Glover**

February, 1987

* Carnegie Mellon University
Pittsburgh, PA

** University of Colorado,
Boulder, CO

This report was prepared in part as part of the activities of the Management Sciences Research Group, Carnegie Mellon University, under Contract No. N00014-85-K-0198 NR 047-048 with the Office of Naval Research and in part by NSF Grant ECS 8601660. Reproduction in whole or in part is permitted for any purpose of the U. S. Government.

Management Sciences Research Group
Graduate School of Industrial Administration
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213

0. Abstract.

We address the problem of finding a minimum weight base B of a matroid when, in addition, each element of the matroid is colored with one of m colors and there are upper and lower bound restrictions on the number of elements of B with color i, for i = 1, 2, ..., m. This problem is a special case of matroid intersection. We present an algorithm that exploits the special structure. When applied to the weighted bipartite matching problem, our algorithm has complexity $O(|E|\,|V| + |V|^2 \log |V|)$. When applied to the problem of finding a minimum weight spanning tree with degree restrictions on the nodes of a stable set, it has complexity $O(|V|^3)$. In both cases, V denotes the node set of the underlying graph, and E denotes its edge set. We also discuss a new relaxation for the traveling salesman problem.

1. Introduction.

Let E be a finite set, M a matroid of rank r defined on E, and $w: E \to R$ a weight function. The weight of a subset $F \subseteq E$ is defined by $w(F) = \Sigma\{w(e) : e \in F\}$. Let $E_1, E_2, ..., E_m$ be a partition of E into nonempty subsets. To each E_i assign two integers l_i and u_i ($l_i \le u_i$). We address the problem of finding a base B of M which will

minimize w(B)

subject to

$$l_i \le |B \cap E_i| \le u_i \quad \text{for } i = 1, 2, ..., m.$$

This problem is denoted by (P).

It is known that the set

$$B^* = \{X \subseteq E : l_i \le |X \cap E_i| \le u_i,\ i = 1, 2, ..., m,\ |X| = r\}$$

is the family of bases of a matroid M', which has rank r. Such a matroid is called a generalized partition matroid. Thus, (P) is the problem of finding a minimum weight common basis of M and M', if one exists.

This problem has been solved in polynomial time by Edmonds ([E1] and [E2]). The algorithm presented here takes advantage of the special structure of the matroid M' and works on an auxiliary digraph with m nodes, instead of the n = |E| nodes required by the general matroid intersection algorithms.

This leads to a complexity of $O(r(nc + m \log m))$, where c is the complexity of circuit finding. By circuit finding we mean the following: Given independent set I and $e \in E - I$, list the elements of the unique circuit in $I + e$, or show that $I + e$ is independent. This complexity is to be compared with the $O(nr(r + c + \log n))$ required by the general algorithm (see Brezovec, Cornuéjols, and Glover ([BCG])).

To make this paper self-contained, we state without proofs some of the results of [BCG]. This is done in Section 2. The algorithm for solving (P) is presented in Section 3.

Finally, in Sections 4 and 5, we provide two examples of how our algorithm may be applied to classical combinatorial optimization problems. For the weighted bipartite matching problem on a graph $G = (V, E)$, we get an algorithm of complexity $O(|V| |E| + |V|^2 \log |V|)$, which is the best known for weighted bipartite matching (see Fredman and Tarjan [FT]). Note that the general matroid intersection algorithm only gives a bound of $O(|E| |V|^2)$.

In section 5, we consider the case where M is the graphic matroid associated with a graph $G = (V, E)$. Assign each edge of G a weight, and let $V^* = \{v_1, v_2, \ldots, v_{m-1}\}$ denote a stable set of G (a set of pairwise nonadjacent nodes). Let color class i consist of the edges incident with node v_i for $i = 1, 2, \ldots, m-1$, with color class m containing all remaining edges. A minimum weight spanning tree of G must be found with the property that, for $i = 1, 2, \ldots, m$, the number of edges with color i belongs to the interval $[l_i, u_i]$. This is of complexity $O(|V|^3)$. Note that, here as well, the general matroid intersection algorithm only gives a bound of $O(|E| |V|^2)$. If $l_i = u_i = 2$ for $i = 1, 2, \ldots, m-1$, the complexity can be further reduced to $O(m |V|^2)$, and such degree restricted spanning trees can be used to strengthen the 1-tree relaxation of Held and Karp for the Traveling Salesman Problem. We also show how this relaxation can be further strengthened and generalized to the case where V^* is not a stable set. In this case the color classes are defined in terms of *net stars* instead of stars.

2. Preliminary results

Let M be a matroid defined on the element set E, and let B be a base of M. Consider $\beta \subseteq B$ and $\beta' \subseteq E - B$ such that $|\beta| = |\beta'|$. We say that (β, β') is a **B-swap** if $B - \beta + \beta'$ is a base of M. We call $m(\beta, \beta')$ a **matching** if it represents a one-to-one mapping of β onto β'; $m(\beta, \beta')$ is called a **B-matching** if every (e, e') in $m(\beta, \beta')$ is a B-swap.

Remark 1. [BCG, Lemma 2]. Let B and B' be two bases of M, and let $\beta = B - B'$ and $\beta' = B' - B$. Then there is at least one B-matching of (β, β').

Remark 2. [BCG, Lemma 3]. Let B be a base of M, and consider $\beta \subseteq B$ and $\beta' \subseteq E - B$ where $|\beta| = |\beta'|$. If $m(\beta, \beta')$ is a B-matching but (β, β') is not a B-swap, then there also exists a B-matching $m'(\beta, \beta')$ different from $m(\beta, \beta')$.

Consider two matroids M_1 and M_2 defined on the same finite element set E, and let $I \subseteq E$ be a k-intersection, that is, let I be independent in M_1 and M_2 such that $|I| = k$. Let $w: E \rightarrow R$ be a weight function. The problem of finding a minimum weight k-intersection is denoted (P_k). To solve (P_k), define a bipartite auxiliary digraph $G(I) = (V, V', A)$ as follows. Construct a node in V for each $e \in I$ and a node in V' for each $e' \in E - I$. For each $e \in I$, we construct an arc (e, e') with weight $w(e, e') = w(e') - w(e)$ provided $I - e + e'$ is independent in M_1. Similarly, for each $e' \in E - I$, we construct an arc (e', e) with weight $w(e', e) = 0$ provided $I - e + e'$ is independent in M_2.

Let C be a dicycle in G(I). We define the weight of this dicycle to be the sum of the weights of the arcs in the dicycle. Thus, $w(C) = w(\Sigma') - w(\Sigma)$, where Σ and Σ' represent the nodes of C in V and V', respectively.

Theorem 1. [BCG, Theorem 3]. Assume I is a k-intersection.

(i) I is optimal for (P_k) if and only if there are no negative weight dicycles in G(I).

(ii) Let C be a negative weight dicycle in G(I) with no negative dicycle

on a subset of its nodes, and let Σ and Σ' be the nodes of C in V and V', respectively. Then $I' = I - \Sigma + \Sigma'$ is a k-intersection such that $w(I') < w(I)$.

This theorem leads to an algorithm for solving (P_k), given an initial k-intersection I: use G(I) to find an improved solution; continue until a G(I) is found which has no negative dicycles.

We modify this technique to develop a dual algorithm. Consider a set A of k artificial elements, where $E \cap A = \emptyset$. Let $F = E + A$ and define two matroids $M_1(F)$ and $M_2(F)$ on the element set F as follows. For $i = 1, 2$, $I \subseteq E$ and $J \subseteq A$, the set $I + J$ is independent in $M_i(F)$ if and only if I is independent in M_i. The problem (P_k) relative to these new matroids is denoted $(P_k|\ F)$. In order to define $(P_k|\ F)$ completely we need to assign weights to the artificial elements. By giving them large negative weights, we can guarantee that $I = A$ is optimal for $(P_k|\ F)$.

Define the digraph Sp(I) from G(I) by splitting one of the nodes arising from an artificial element, say $\alpha \in A$, into a source node s and a destination node d. Let the arcs out of s in Sp(I) be those out of α in G(I) and the arcs into d in Sp(I) be those into α in G(I). The following result provides the basis for an algorithm to solve (P_k).

Theorem 2. [BCG, Theorem 5]. Let I be optimal for $(P_k|\ H)$, where $E \subseteq H \subseteq F$, and let Sp(I) be obtained from G(I) by splitting an artificial element α into s and d.

(i) Problem $(P_k|\ H - \alpha)$ has no solution if and only if there is no s-d dipath in Sp(I).

(ii) Let P be a shortest s-d dipath such that every s-d dipath defined on a subset of its nodes has a strictly larger weight, and let Σ and Σ' be the nodes of P in V and V', respectively. (s and d both give rise to α in Σ.) Then $I' = I - \Sigma + \Sigma'$ is optimal for $(P_k|\ H - \alpha)$.

We use this to solve (P_k) as follows. Start with $H = F$ and the set $I = A$.

Construct G(I) and then Sp(I) by splitting one node arising from an artificial element α. Find a shortest dipath from the resulting source s to destination d, with the added condition that every s-d dipath on a proper subset of its nodes has a strictly larger weight. Then the set I' defined in Theorem 2(ii) is optimal for $(P_k|\ H - \alpha)$. Repeat this process with I' in place of I and $H - \alpha$ in place of H, until all artificial elements have been split. When this occurs, we have an optimal solution for (P_k).

We close this section with a result that will be useful in the proof of Theorem 6.

Theorem 3. [BCG, Corollary]. Let $e \in V$ and $e' \in V'$.

(a) If P is an e-e' dipath in Sp(I) and $I - \Sigma + \Sigma'$ is a dependent set in M_1, then there is an e-e' dipath P_0 in Sp(I) such that the intermediate nodes of P_0 constitute a proper subset of those of P, and $w(P_0) \leq w(P)$.

(b) If P is an e'-e dipath in Sp(I) and $I - \Sigma + \Sigma'$ is a dependent set in M_2, then there is an e'-e dipath P_0 in Sp(I) such that the intermediate nodes of P_0 constitute a proper subset of those of P, and $w(P_0) \leq w(P)$.

3. The Algorithm

In this section we show how to solve (P). The question of finding a first basis B is not addressed until later in the section.

For convenience, we index each element of E to identify the subset or **state** of the partition $E_1, E_2, ..., E_m$ to which it belongs. Thus, e_i and f_j are elements of E_i and E_j, respectively. Also, the symbol ' is used to denote whether or not an element is in B, so that $e_i \in B$ but $e_j' \in E - B$.

We define the **state graph of B**, denoted $S(B) = (N, A)$, as the following directed graph. The node set is given by $N = \{v_1, v_2, ..., v_m\}$, where node v_i corresponds to state i, $i = 1, 2, ..., m$. For each e_i in B we construct an arc $a(e_i, e_j')$ with weight $w(a(e_i, e_j')) = w(e_j') - w(e_i)$ directed from node v_i to node v_j provided (e_i, e_j') is a B-swap. We call these the **forward arcs** of S(B). *Note that, in general, S(B) is a multigraph and can have loops.* For each e_j' in E - B, construct $a(e_j', e_l)$ from v_j to v_l, with weight 0, provided $B - e_l + e_j'$ does not violate the cardinality conditions of E_l and E_j. We call these the **backward arcs** of S(B). Note that the existence of a backward arc from v_j to v_l indicates that we can exchange <u>any</u> element e_l in $B \cap E_l$ for <u>any</u> element e_j' in $(E - B) \cap E_j$ without violating the cardinality conditions on $B \cap E_l$ and $B \cap E_j$. All such backward arcs $a(e_j', e_l)$ appear in parallel in S(B).

We could have constructed S(B) using G(B), the bipartite auxiliary digraph discussed in the previous section, as follows. Let $M_1 = M$ and let M_2 be the matroid whose bases are elements of the set

$$B^* = \{X \subseteq E:\ l_i \le |X \cap E_i| \le u_i,\ i = 1, 2, ..., m,\ |X| = r\}.$$

Note that (P_r) defined by M_1 and M_2 is equivalent to problem (P). Construct G(B). Consider, for $i = 1, 2, ..., m$, the set of nodes corresponding to E_i and identify this set as the node v_i. Thus, each arc (e_i, e_j') or (f_i', f_j) becomes an arc joining v_i to v_j, and this new graph is clearly S(B).

Let C be a dicycle of S(B). In this paper, dicycles can have repeated nodes, but no repeated arcs. Note that this does not exclude the possibility of repeated elements of E. We say that $\beta \subseteq B$ and $\beta' \subseteq E - B$ **give rise** to C if

$\beta = \{e_i \in B : a(e_i, e_j')$ is a forward arc of C$\}$, and

$\beta' = \{e_j' \in E - B : a(e_i, e_j')$ is a forward arc of C$\}$.

We define the weight of this dicycle, w(C), to be the sum of the weights of the arcs in the dicycle. Then $w(C) = w(\beta') - w(\beta)$, where $\beta \subseteq B$ and $\beta' \subseteq E - B$ are the sets of elements which give rise to the dicycle. Dipaths of S(B) have similar definitions, with the following exception. If the last arc of a dipath P is a backward arc, define

$\beta = \{e_i \in B : a(e_i, e_j')$ is a forward arc of P

or $a(e_k', e_i)$ is the last arc of P$\}$.

Note that, if $\beta \subseteq B$ and $\beta' \subseteq E - B$ give rise to an element disjoint dicycle C in S(B), then they also give rise to a dicycle of the form

$$\{a(e_i, e_j'), a(e_j', e_k), \ldots, a(e_l, e_q'), a(e_q', e_i)\}. \qquad (*)$$

There are four ways in which C could fail to have this form: C could contain, as subsequences,

(i) consecutive forward arcs $\{a(e_i, e_j'), a(f_j, e_k')\}$,

(ii) consecutive backward arcs $\{a(e_i', e_j), a(f_j', e_k)\}$,

(iii) a forward-backward pair with different elements from E - B, $\{a(e_i, e_j'), a(f_j', e_k)\}$, or

(iv) a backward-forward pair with different elements from B, $\{a(e_j', f_k), a(e_k, e_l')\}$.

In case (i) the loop $a(e_j', f_j)$ must be a backward arc in S(B) since swapping these two elements in B will not change $|B \cap E_j|$. So replace the consecutive forward arcs in C with $\{a(e_i, e_j'), a(e_j', f_j), a(f_j, e_k')\}$. (Recall that we only require dicycles to be arc-disjoint; so loops are permitted.)

The result of the consecutive backward arcs in case (ii) is that the cardinality of E_i increases by one in $B - \{e_j, e_k\} + \{e_i', f_j'\}$, with the cardinality of E_k decreasing by one. E_j realizes no change in cardinality in from this string. Thus, $a(e_i', e_k)$ must be a backward arc in S(B), and we can replace the consecutive backward arcs in case (ii) with the single backward arc $a(e_i', e_k)$.

In case (iii), the backward arc $a(f_j', e_k)$ can be replaced with the

parallel arc $a(e_j', e_k)$. Similarly, $a(e_j', f_k)$ can be replaced with $a(e_j', e_k)$ in case (iv).

Since these steps (which result in a dicycle that has the form (*)) only affect the backward arcs in the dicycle, β and β' are unchanged.

The following result is immediate.

Lemma 1. There is a one-to-one correspondence between element-disjoint dicycles in S(B) having the form (*) and node-disjoint dicycles in G(B).

Theorem 4. Assume B is feasible for (P).

(i) B is optimal for (P) if and only if there are no negative dicycles in S(B).

(ii) Let C be a negative dicycle in S(B) with no negative dicycle on a subset of the elements $\beta + \beta'$ which give rise to C. Then $B' = B - \beta + \beta'$ is a base such that $w(B') < w(B)$.

Proof. This result follows directly from Lemma 1, Theorem 1, and the following observation. If a negative dicycle in S(B) has a repeated element, that dicycle induces two new dicycles, one of which must be negative. Thus, if there is a negative dicycle, there is a negative dicycle without repeated elements. //

Theorem 4 shows the equivalence of solving (P) and finding negative weight dicycles in S(B) with the property that no negative dicycle exists on a subset of the elements which give rise to it. This gives the basis for a primal algorithm: start with a feasible B and use S(B) to find an improved solution. Continue in this manner until we find an S(B) with no negative dicycles.

We briefly discuss the complexity issues of this algorithm: Firstly, one must be able to find a negative dicycle in S(B) with the required property. Also, the number of iterations needed to reach the optimal solution can be relatively large; even though each iteration yields a better solution, the improvement may be slight. For these reasons the complexity is high, and we do not pursue our investigation of this primal algorithm. (We do note, however, that this approach may have merits in the context of sensitivity analysis: Given an optimal solution for a certain set of weights,

reoptimizing on a perturbed set of weights using this method may be more efficient in practice than starting from scratch, as other algorithms for matroid intersection would require.)

Instead we use these results to develop a dual algorithm. First, using a modification of the greedy algorithm, we can find an initial solution B_0. Start with $B_0 = \varnothing$ and, at each iteration, add an element $e \in E - B_0$ to B_0 if

(i) $B_0 + e$ is independent in M,

(ii) e has the smallest weight of all such $f \in E - B_0$, and

(iii) $|(B_0 + e) \cap E_i| \leq u_i$ for $i = 1, 2, ..., m$.

Stop when either no such e can be found, $|B_0| = r$, or, for the e chosen,

$$r - |B_0 + e| < \Sigma\{\max(l_i - |(B_0 + e) \cap E_i|, 0): \ i = 1, 2, ..., m\}.$$

Violating this last condition would result in a B_0 which could not satisfy both $|B_0| = r$ and all lower bound conditions. At the termination of this procedure we clearly have the least weight independent set with $|B_0|$ elements.

If care has been taken to delete loops of the matroid M from E, B_0 will have at least one element. If $|B_0| = r$, (P) is solved. Otherwise, consider a set A of $r - |B_0|$ artificial elements, where $E \cap A = \varnothing$. We assign to each artificial element a state as follows. Extend E_1 by unioning $\max(0, l_1 - |B_0 \cap E_1|)$ artificial elements. Then extend $E_2, E_3, ..., E_m$ similarly. Any remaining artificial elements can then be unioned to any of the states, as long as the resulting states satisfy $|(B_0 + A) \cap E_i| \leq u_i$. Note that, if this cannot be done, (P) is infeasible.

Let $F = E + A$ and define a matroid M(F) on the set F as follows: for $I \subseteq E$ and $J \subseteq A$, $I + J$ is independent in M(F) if and only if I is independent in M. The problem (P) relative to M(F) will be denoted (P| F); we now wish to find the least weight independent subset B of F such that $l_i \leq |B \cap E_i| \leq u_i$, for $i = 1, 2, ..., m$, and $|B| = r$. Note that, if we define the matroids M_1 and M_2 as at the beginning of this section and $M_1(F)$ and $M_2(F)$ in the obvious way, then (P| F) is equivalent to (P_r| F) defined by $M_1(F)$ and $M_2(F)$.

To define (P| F) completely we need to assign weights to the artificial elements. If we give each artificial a large positive weight, we can guarantee that, if (P) is feasible, then an optimal solution to (P| F) contains no artificial elements and is, hence, optimal for (P) as well. Thus, we can use $B_0 + A$ as a starting solution, and solve (P| F) using the primal algorithm whose basis is given earlier in this section.

However, we have a different algorithm in mind. We will give large negative weights to all artificial elements. Thus, the initial solution $B = B_0 + A$ will be optimal for (P| F).

Define the digraph S*(B) from S(B) by creating two nodes, a source node s and a destination node d, from an artificial element, say $\alpha \in A$. We also allow s and d to denote the resulting elements. Assume α was assigned to E_i. Replace arcs of the form $a(\alpha, e_j')$ in S(B) with arcs $a(s, e_j')$ in S*(B), and replace arcs of the form $a(e_l', \alpha)$ in S(B) with $a(e_l', d)$ in S*(B).

Note that we could have constructed S*(B) by forming Sp(B) from G(B), and then shrinking groups of nodes (excluding s and d) corresponding to each of the states into the nodes $v_1, v_2, \ldots, v_m$. Thus the following theorem is a direct result of Theorem 2.

Theorem 5. Let B be optimal for (P| H), where $E \subseteq H \subseteq F$, and let S*(B) be obtained from S(B) by creating s and d from artificial element α.

(i) Problem (P| H - α) has no solution if and only if there is no s-d dipath in S*(B).

(ii) Let P be a shortest s-d dipath such that every s-d dipath defined on a subset of the elements which give rise to P has a strictly larger weight, and let β and β' be the elements of B and H - B, respectively, that give rise to the nodes of P. (s and d both give rise to α in β.) Then $B' = B - \beta + \beta'$ is optimal for (P| H - α).

Theorem 5 provides the basis for our algorithm. Start with H = F and $B = B_0 + A$, which is optimal for (P| F). Construct S(B) and then S*(B) by creating s and d from artificial element α. Find a shortest s-d dipath P such that every dipath on a subset of the elements which give rise to P has a strictly larger weight. Then the set B' defined in Theorem 5(ii) is optimal

for (P| H - α). Repeat this process with B' in place of B and H - α in place of H, until the resulting B' contains no artificial elements. At this step, B' is optimal for (P).

Note that the arc weights of $S(B_0 + A)$ are all nonnegative. To see this, first consider arcs of the form $a(\alpha, e_j')$, where α is artificial. Since $w(\alpha)$ has been chosen small enough, $w(a(\alpha, e_j')) = w(e_j') - w(\alpha) > 0$. Now consider an arc $a(e_i, e_j')$, where $e_i \in B_0$. Since this arc is in $S(B_0 + A)$, we have $B_0 - e_i + e_j'$ independent in M. But then $w(e_i) \leq w(e_j')$ by requirements (i) and (ii) of our modified greedy approach in the construction of B_0; so $w(a(e_i, e_j')) \geq 0$. Finally, when e is not in $B_0 + A$, $w(a(e, f)) = 0$.

All arc weights nonnegative in $S(B_0 + A)$ implies the same for $S^*(B_0 + A)$; so no shortest s-d dipath will repeat nodes (or, hence, elements). Thus, for all pairs of nodes, we only need to consider the shortest of the arcs joining v_i to v_j when searching for the shortest s-d dipath, and we can apply Dijkstra's algorithm to find a shortest s-d dipath P. Furthermore, the requirement that every dipath on a subset of the elements which give rise to P has a strictly larger weight can be obtained by adding a small positive ε to all the arc weights of $S^*(B_0 + A)$.

We now show that nonnegative weights on the arcs of S(B) can be preserved throughout the algorithm. Define a variable $D(v_i)$ associated with every node of the digraph $S^*(B)$ and a reduced weight

$$w'(a(e_i, e_j')) = w(a(e_i, e_j')) + D(v_i) - D(v_j)$$

associated with each forward arc of S(B). Similarly, for each backward arc define

$$w'(a(e_k', e_l)) = D(v_k) - D(v_l).$$

Note that, in terms of the reduced weights, the length of an s-d dipath P is $w'(P) = w(P) + D(s) - D(d)$ since, for any intermediate node v_i, the variable $D(v_i)$ cancels out on the two arcs of P that contain v_i. Since D(s) - D(d) is a constant that does not depend on P, it is equivalent to solve the shortest dipath problem in $S^*(B)$ with the reduced weights w' instead of the original weights w. We provide a choice for the variables $D(v_i)$ that guarantees

nonnegative reduced weights from one iteration to another in the next theorem.

Theorem 6. Let B be optimal for (P| H) where $E \subseteq H \subseteq F$, and assume that an s-d dipath exists in $S^*(B)$. Set $D(v_i)$ to be the length of a shortest s-v_i dipath in $S^*(B)$, for i = 1, 2, ..., m. Let B' be as defined in Theorem 5(ii). Then, the reduced weights $w'(a(e_i, e_j')) = w(a(e_i, e_j')) + D(v_i) - D(v_j)$ and $w'(a(e_k', e_l)) = D(v_k) - D(v_l)$ are nonnegative for every forward arc $a(e_i, e_j')$ and every backward arc $a(e_k', e_l)$ of the digraph S(B') for problem (P| H - α).

It was noted above that every dicycle in S(B) can be assumed to have the form (*). Using the same argument, we can assume every s-d dipath in $S^*(B)$ to have the form

$$P = \{a(s, e_i'), a(e_i', e_j), a(e_j, e_k'), ..., a(e_l', d)\}. \qquad (**)$$

Note that, since the only arcs affected by changing an s-d dipath to one of the form (**) are backward arcs, putting an s-d dipath into this form does not change the weight of the dipath.

If an s-d dipath is of the form (**), the elements which give rise to P are all elements represented in an arc of P since these are exactly the elements which are represented in the forward arcs of P. For the proof of Theorem 6 we assume that the shortest s-d dipath has been put in the form (**). We also need the following remark, which is a direct result of Theorem 3.

Remark 3. Let P be a shortest s-d dipath in $S^*(B)$ such that every s-d dipath of the form (**) on a subset of the elements which give rise to P has strictly larger weight, and let P_o be some subpath of P from s to another node. Let β_o and β_o' be the sets of elements which give rise to P_o.

(a) If the last arc on P_o is a forward arc, then $B - \beta_o + \beta_o'$ is independent in M.

(b) If the last arc on P_o is a backward arc, then $B - \beta_o + \beta_o'$ satisfies the cardinality conditions on each E_i.

Proof of Theorem 6. First we show that the choice for $D(v_i)$ gives nonnegative reduced weights on the arcs of S(B). Consider $a(e_i, e_j')$ in S(B), and suppose $w(a(e_i, e_j')) + D(v_i) - D(v_j) < 0$. This gives

$$D(v_i) + w(a(e_i, e_j')) < D(v_j),$$

which implies that the shortest s-v_i dipath together with the arc $a(e_i, e_j')$ gives a shorter distance from s to v_j than $D(v_j)$, a contradiction. Similarly, for each backward arc $a(f_i', f_j)$ in S(B), $w(a(f_i', f_j)) + D(v_i) - D(v_j) \geq 0$. Thus, our choice of $D(v_i)$ gives nonnegative reduced weights on the arcs of S(B).

Let P be the shortest s-d dipath that gives rise to B' in Theorem 5(ii), and let P_i be the subpath of P from s to some node v_i. Let $\beta_1(i)$ and $\beta_1'(i)$ be the sets of elements of B and E - B, respectively, represented by the forward arcs of P_i, and let $\beta_2(i)$ and $\beta_2'(i)$ be the corresponding sets of elements represented by the backward arcs of P_i. By Remark 4, $B_1(i) = B - \beta_1(i) + \beta_1'(i)$ is independent in M, and $B_2(i) = B - \beta_2(i) + \beta_2'(i)$ satisfies the cardinality conditions on each E_i. For ease of notation we allow v_i to represent s and d, as well as the nodes representing states $E_1, E_2, ..., E_m$.

We construct a digraph N_i as follows. The node set of N_i is that of S(B). If $B_1(i) - e_j + e_k'$ is independent in M, then $a(e_j, e_k')$ is a forward arc of N_i; if $B_2(i) - e_j + e_k'$ satisfies the cardinality conditions of each E_i, then $a(e_k', e_j)$ is a backward arc of N_i. Note that, in the latter case, we call (e_k', e_j) a $B_2(i)$-swap with respect to the partition matroid.

We will show that this choice of $D(v_i)$ gives nonnegative reduced weights on the arcs of N_i by induction on the nodes of P, starting from $v_i = s$. Note that when we reach $v_i = d$, we will have $N_i = S(B')$ for $(P| H - \alpha)$, and the theorem will be proved.

When $v_i = s$, $N_i = S(B)$, and the result has been proved above. Let v_i be a node of P for which the result holds, and let v_j be the node following v_i in P. There are two cases to consider, as the arc joining v_i to v_j can be a forward arc or a backward arc.

Suppose the arc joining v_i to v_j in P is a forward arc $a(e_i, e_j')$. In this case the backward arcs of N_j are the same as those of N_i; hence, they have nonnegative reduced weights by induction. Thus, we only need to show $D(v_q) - D(v_p) \leq w(e_q') - w(e_p)$ for every $B_1(j)$-swap (e_p, e_q'). Note that if (e_p, e_q') is also a $B_1(i)$-swap we are done. So only consider $B_1(j)$-swaps which are not $B_1(i)$-swaps.

If $(e_p, e_q') = (e_j', e_i)$, then

$$D(v_j) = D(v_i) + w(e_j') - w(e_i) \qquad (\dagger)$$

since $a(e_i, e_j')$ is on the shortest s-d dipath P, and the desired inequality is immediate.

For the remaining cases, consider $B^* = B_1(i) - e_i + e_j' - e_p + e_q'$, which is independent in M. Suppose $e_p = e_j'$ and $e_q' \neq e_i$. Then $B^* = B_1(i) - e_i + e_q'$ and (e_i, e_q') is a B_i-swap. Therefore, $w(e_q') - w(e_i) + D(v_i) - D(v_q) \geq 0$ by induction. Combining this inequality with $(\dagger)$ and $e_p = e_j'$, we get $w(e_q') - w(e_p) + D(v_p) - D(v_q) \geq 0$, and the result follows. A similar argument applies when $e_p \neq e_j'$ and $e_q' = e_i$.

Finally, suppose $e_p \neq e_j'$ and $e_q' \neq e_i$. Since (e_p, e_q') is not a $B_1(i)$-swap, yet B^* is a base, it follows from Remarks 1 and 2 that both (e_p, e_j') and (e_i, e_q') are $B_1(i)$-swaps. Thus, $w(e_j') - w(e_p) + D(v_p) - D(v_j) \geq 0$, and $w(e_q') - w(e_i) + D(e_i) - D(e_q) \geq 0$. Summing these inequalities and applying $(\dagger)$, we get $w(e_q') - w(e_p) + D(v_p) - D(v_q) \geq 0$.

Now suppose the arc joining v_i to v_j in P is a backward arc $a(f_i', f_j)$. Then $B_1(j) = B_1(i)$, and the forward arcs of N_j are the same as those of N_i. Hence, they have nonnegative reduced weights by induction. Thus, we only need to show $D(v_p) - D(v_q) \geq 0$ for every $B_2(j)$-swap (e_p', e_q). Note that if (e_p', e_q) is also a $B_2(i)$-swap we are done by induction. So only consider $B_2(j)$-swaps which are not $B_2(i)$-swaps.

If $(e_p', e_q) = (f_j, f_i')$, then

$$D(v_j) = D(v_i) \qquad (\dagger\dagger)$$

since $a(f_j', f_i)$ is on the shortest s-d dipath P, and the desired inequality is immediate.

For the remaining cases consider $B^{**} = B_2(i) - f_j + f_i' - e_q + e_p'$, which satisfies the cardinality conditions for each E_i since (e_p', e_q) is a $B_2(j)$-swap. Suppose $e_p' = f_j$ and $e_q \neq f_i'$. Then $B^{**} = B_2(i) - e_q + f_i'$, and (f_i', e_q) is a $B_2(i)$-swap. Hence, $D(v_i) - D(v_q) \geq 0$ by induction. This and equation (††) show that $D(v_p) - D(v_q)$. The proof is similar when $e_p' \neq f_j$ and $e_q = f_i'$.

Finally, suppose $e_p' \neq f_j$ and $e_q \neq f_i'$. Since (e_p', e_q) is not a $B_2(i)$-swap, yet B^{**} is independent in the partition matroid, it follows from Remarks 1 and 2 that both (e_p', f_j) and (f_i', e_q) are $B_2(i)$-swaps. Thus, induction yields $D(v_p) - D(v_j) \geq 0$ and $D(v_i) - D(v_q) \geq 0$. Summing these inequalities and applying (††), we get $D(v_p) - D(v_q) \geq 0$. //

We now summarize the complexity of the algorithm. O(r) iterations will be needed, one iteration for the removal of each artificial element in the initial solution. In each iteration we need to construct S(B), find a shortest s-d dipath, and update the variables $D(v_i)$. Recall that $|E| = n$, and c represents the time required to find the circuit of $I + e'$ in M (or show that none exists), where $I \subseteq E$ is independent.

To construct S(B) for the problem (P| H), $E \subseteq H \subseteq F$, we solve the following circuit recognition problem for each $e' \in H - B$: Check whether $B + e'$ is dependent in M(H), and, if it is, find the unique circuit of $B + e'$. This task requires time c for each $e' \in H - B$. So the complexity of constructing S(B) is O(nc). In some instances it is possible to speed up the construction of S(B), as not all parallel arcs need to be constructed, but only the shortest. An example of this will be given in Section 5.

Using the fast implementation of Fredman and Tarjan (see [FT]), the complexity of finding a shortest s-d dipath by Dijkstra's algorithm is at most $O(\rho + m \log m)$, where ρ is the cardinality of the set of arcs from $S^*(B)$ used. Recall that we do not need loops and only consider the shortest in each group of parallel arcs to find a shortest s-d dipath.

Note that the variables $D(v_i)$ needed in Theorem 6 are actually computed in the course of finding a shortest s-d dipath. Thus, no extra computations are needed. Also, nc will be larger than ρ. Therefore, the overall complexity of the algorithm is $O(r(nc + m \log m))$.

4. Weighted bipartite matching.

Consider a graph $G = (V, E)$ and a weight function $w: E \rightarrow R$. The matching problem consists of finding a minimum weight subset $F \subseteq E$ such that each node of V is incident with exactly one member of F. In this section we assume that G is a bipartite graph where the partition $V_1 \cup V_2 = V$ is such that $|V_1| = |V_2|$. The weighted bipartite matching problem, also called the assignment problem, is a special instance of problem (P) where the partition of E is induced by the nodes of V_2. Specifically, $e \in E_i$ if and only if it is incident with node $v_i \in V_2$. For each i, $l_i = u_i = 1$. The matroid M is also a partition matroid, namely the one induced by the node set V_1.

For this problem $r = |V|/2$, $n = |E|$, and $m = |V|/2$. To obtain the complexity of our algorithm for this problem, we have to determine the complexity of constructing S(B). Note that the circuits of M have length 2; so the number of forward arcs of S(B) is at most |E|. Similarly, there are at most |E| backward arcs.

Thus, the construction of S(B) requires time O(|E|). Since the Fredman Tarjan algorithm finds the minimum length dipath in time $O(|E| + |V| \log |V|)$, the overall complexity is $O(|V|\,|E| + |V|^2 \log |V|)$.

Note that our algorithm is closely related to the usual augmenting path algorithm. Although this algorithm and the associated bound are not new, it is interesting that they are obtained using the state graph approach. Recall that the general matroid intersection algorithm only gives the bound $O(|V|^2\,|E|)$.

5. Minimum spanning trees.

Let G = (V, E) be an undirected graph, and let each edge of G be assigned a weight. Let $V^* = \{v_1, v_2, ..., v_{m-1}\}$ be a stable set of G; that is, no edge of E has both endpoints in V^*. For i = 1, 2, ..., m-1, let E_i denote the edges which are incident with node v_i in G, and let E_m be the remaining edges. For each i assign two integers l_i and u_i, where $l_i \le u_i$. The minimum spanning tree problem with restrictions on the number of edges in each E_i is an instance of (P): M is the graphic matroid defined on the edge set E, and the partition matroid is given by the restriction that the number of edges from state i belongs to the interval $[l_i, u_i]$.

An interesting feature of this instance is that, by using the special structure of the tree, we do not need to construct the entire state graph S(B). However, our method requires that we work with spanning trees. So, instead of simply extending the graphic matroid M to M(F) as described in Section 3, create an artificial edge in G for each artificial element so that the resulting first solution, $B_0 + A$, is a spanning tree in the extended graph $G^* = (V, E + A)$.

Define the **star** of a node to be the set of edges incident with that node. Suppose the nodes are ordered from 1 to |V|. For node 1 define its **net star** to be its star. Then, for i = 2, 3, ..., |V|, define the **net star** of node i to be its star, excluding edges which appear in the stars of nodes 1, 2, ..., i-1.

Let V^* be nodes 1 to m-1, and let the remaining nodes be nodes m to |V|. Then the net star for each node v_i in V^* is E_i, and E_m consists of the net stars for $V - V^*$. Begin by sorting the edges in each net star in ascending order by weight. At worst, this work is O(|E| log |V|). Note that we need to sort the edge weights only once since the reduced weights only add a constant to all the edges in the same net star. At each iteration of our algorithm, in the construction of each new state graph, proceed as follows:

1. Set up (or update) the current basis tree with a root and predecessor indexing.

2. For i = 1, 2, ..., |V|, start with all nodes unlabeled, and perform the following labeling procedure, examining nonbasic edges in node i's net star in ascending weight order.

(a) Label all nodes in the predecessor path from node i to the root with an x, and label i with a y.

(b) Let j be the other endpoint of the nonbasic edge (i, j) currently under examination. Trace the predecessor path for j until reaching a labeled node k, where, possibly, k = j. The edges between k and j in the tree have never been examined for the current node i. Check each as an edge to swap with (i, j), either to create a forward arc in the state graph, or to replace an existing parallel one of larger weight. As this check occurs, label the nodes between nodes k and j in the tree, excluding k, with a y.

(c) If node k has a y label, repeat (b) with the next edge in node i's net star. Otherwise, k has an x label and, hence, lies on the predecessor path from node i to the root. In this case, trace from node k to node i, and consider swapping each edge encountered with (i, j), stopping when encountering a node with a y label. When this occurs, label each node encountered with a y.

Note that all nodes of every cycle formed with an added edge will be labeled with a y, although possibly not all of the cycle will be examined on the current step, since some edges may have been examined before. Also, at every step, the edges examined will be exactly the edges of the current cycle that have never been examined before. Finally, because we examine the nonbasic edges of each net star in order by increasing weight, we find the minimum weight arc in S(B) from v_i to v_j for each pair (v_i, v_j) of states.

We now consider the complexity of our algorithm. During the pass through the net star for node i, which has at most |V| - i edges, the work of examining the edges to be dropped will be on the order of the number of nodes in the tree, not previously labeled, which are now labeled. So, O(|V|) work is required for each nonempty net star, which yields $O(|V|^2)$ work total.

Note that the complexity due to the Fredman-Tarjan shortest dipath algorithm, at most $O(m^2)$, is dominated by the $O(|V|^2)$ required to construct each state graph. The graphic matroid has rank at most |V| - 1. Hence, the overall complexity is $O(|V|^3)$.

In the special case where $l_i = u_i = 2$ for $i = 1, 2, ..., m-1$, and $l_m = 0$ and $u_m = |V| - 2m + 1$, the greedy phase of finding a starting solution B_0 can substantially reduce the number of iterations needed. For $i = 1, 2, ..., m-1$, add a constant c_i to the weights of the edges incident with node v_i in V^*. We assign large enough values to the constants so that the greedy phase will first choose $|V| - 2m + 1$ edges in E_m to construct the spanning tree. Let w be the weight of the last edge added. For $i = 1, 2, ..., m-1$, we adjust the constant c_i so that the minimum weight edge incident with v_i has weight w and, therefore, can be chosen next in the greedy phase.

This extended greedy solution B_0 contains $|V| - m$ edges, so only m-1 artificial elements are needed. They can be chosen so that $B_0 + A$ is a tree having degree 2 at each of the nodes of V^*. Furthermore, if the weights of the artificial elements are chosen small enough, $B_0 + A$ is an optimum solution of $(P| E + A)$ and, therefore, a valid start for our dual algorithm. Hence, only m - 1 iterations of the algorithm are needed. This yields an algorithm of complexity $O(m |V|^2)$.

In this case (P) is the problem of finding the minimum weight spanning tree in G such that every node in V^* is adjacent to exactly two edges of the tree. Note that such a spanning tree is a relaxation of a hamiltonian path, namely, a simple path which contains each node.

Consider the following variant of spanning trees. A **1-tree** is a graph with nodes $\{1, 2, ..., |V|\}$ consisting of a spanning tree on $\{2, 3, ..., |V|\}$ together with two edges incident with node 1. The traveling salesman problem on $G = (V, E)$ seeks a minimum weight tour, a cycle which passes through each node exactly once. Observing that a tour is precisely a 1-tree in which each node has degree 2, Held and Karp explored approaches to the traveling salesman problem which involve 1-tree relaxations (see [HK1] and [HK2]).

Our technique permits a generalization of Held and Karp's method to yield a constrained 1-tree as follows. Suppose the nodes are indexed so that $\{1, 2, ..., m-1\}$ is a stable set of G, and let $V^* = \{2, 3, ..., m-1\}$. Then, instead of using a general spanning tree on $\{2, 3, ..., |V|\}$, find a minimum weight spanning tree such that every node in V^* has degree 2. Add the two edges incident with node 1 of minimum weight. Clearly, this constrained 1-tree is still a relaxation of a minimum weight tour. It is tighter than the 1-tree

relaxation, which is obtained when $V^* = \emptyset$.

A still tighter relaxation can be obtained using the net star construction. For each node i, let r_i be the number of edges deleted from its star to create its net star. Then we can stipulate that the spanning tree on {2, 3, ..., |V|} must contain at least max{0, 2 - r_i} and at most 2 edges from the net star of each node $i \geq 2$, which introduces additional constraints to the requirement that every node in V^* has degree 2. This is a valid relaxation of the traveling salesman problem even if {1, 2, ..., m-1} is not a stable set. Note that the complexity of the algorithm in this case is still $O(m\,|V|^2)$.

Finally, consider problem (P) where M is a graphic matroid and E_1, E_2, ..., E_m is a general partition of the edge set. Net stars can still be used within each E_i; therefore, the least weight arc from state i to each of the other states can be found in $O(|V|^2)$. This yields a complexity of $O(m|V|^2)$ for the construction of the state graph and $O(m|V|^3)$ for the overall algorithm. This improves on the complexity of the general matroid intersection algorithm when $|E| > O(m|V|)$. We leave open the question of improving this complexity bound.

References.

[BCG] C. Brezovec, G. Cornuéjols, and F. Glover, "Two algorithms for weighted matroid intersection," *Mathematical Programming* **36** (1986).

[E1] J. Edmonds, "Submodular functions, matroids and certain polyhedra," in: R. Guy, ed., *Combinatorial Structures and their Applications, Proceedings of the Calgary International Conference*, (Gordon and Breach, New York, 1970) 69-87.

[E2] J. Edmonds, "Matroid intersection," *Annals of Discrete Mathematics* **4** (1979) 39-49.

[FT] M.L. Fredman and R.E. Tarjan, "Fibonacci heaps and their uses in improved network optimization algorithms," *Proceedings of the 25th Annual IEEE Symposium on the Foundations of Computer Science* (1984) 338-346.

END

6-87

DTIC